Muckle+: End-to-End Hybrid Authenticated Key Exchanges

Sonja Bruckner[1⋆], Sebastian Ramacher[2], and Christoph Striecks[2]

[1] University of Applied Sciences Upper Austria, Hagenberg, Austria
[2] AIT Austrian Institute of Technology, Vienna, Austria
{firstname.lastname}@ait.ac.at

Abstract. End-to-end authenticity in public networks plays a significant role. Namely, without authenticity, the adversary might be able to retrieve even confidential information straight away by impersonating others. Proposed solutions to establish an authenticated channel cover pre-shared key-based, password-based, and certificate-based techniques. To add confidentiality to an authenticated channel, authenticated key exchange (AKE) protocols usually have one of the three solutions built in. As an amplification, hybrid AKE (HAKE) approaches are getting more popular nowadays and were presented in several flavors to incorporate classical, post-quantum, or quantum-key-distribution components. The main benefit is redundancy, i.e., if some of the components fail, the primitive still yields a confidential and authenticated channel. However, current HAKE instantiations either rely on pre-shared keys (which yields inefficient end-to-end authenticity) or only support one or two of the three above components (resulting in reduced redundancy and flexibility).
In this work, we present an extension of a modular HAKE framework due to Dowling, Brandt Hansen, and Paterson (PQCrypto'20) that does not suffer from the above constraints. While their instantiation, dubbed Muckle, requires pre-shared keys (and hence yields inefficient end-to-end authenticity), our extended instantiation called Muckle+ utilizes post-quantum digital signatures. While replacing pre-shared keys with digital signatures is rather straightforward in general, this turned out to be surprisingly non-trivial when applied to HAKE frameworks (resulting in a significant model change with adapted proof techniques).

Keywords: end-to-end security, hybrid authenticated key exchange, quantum key distribution

1 Introduction

Confidential and authenticated communication channels are a corner stone of today's digital world [DH76,Mau93,BR95]. From user-to-user to server-to-server communication, any data exchanged between any two parties is expected to be

⋆ The work was conducted while at AIT Austrian Institute of Technology.

confidential even in the event of a potentially active man-in-the-middle attack. Ensuring confidentiality between two parties first requires that one can distinguish friend from foe. Specifically, if an adversary can impersonate a party in the system, all confidentiality guarantees are void since in that case the communication with the adversary is secured against outsiders, but the adversary itself may gain access to all data. Therefore, authenticity is a necessary requirement for achieving confidentiality on any level in any system and in the specific context of communication we thus require *end-to-end* authenticity. That is, both parties can directly verify the authenticity of the other party regardless of how many untrusted network links are located between them.

For network protocols on public or untrusted networks, well-established protocols such as Transport Layer Security (TLS) [Res18], IPsec [Kau05], QUIC [IT21], WireGuard [Don17] employ various forms of an end-to-end authenticated key exchange (AKE) [BR95]; on the one hand to authenticate the other peer and on the other hand to establish an ephemeral session key to secure the communication channel. Depending on the concrete application, AKE protocols offer certificate-based authentication, password-based authentication, pre-shared key-based authentication whereas the secret keys are exchanged often using an ephemeral Diffie-Hellman key exchange or – on a more abstract level – with a key exchange using ephemeral key encapsulation mechanism (KEM) keys. Authentication in those protocols may be unilateral, e.g., only the initiator verifies the authenticity of the responder which is the default deployment mode of TLS on the web as the authentication of users is managed on the application layer, or mutual.

End-to-End Authentication Techniques. We will now discuss different techniques to achieve authenticity for key exchange protocols: in a key exchange with pre-shared keys (PSK), both peers are required to agree on a secret key off-channel. This key is then part of the key exchange protocol (e.g., is used as input in the key derivation function to derive the session keys) and only if the key is known, the protocol can be completed successfully. As a folklore consequence, networks with n peers necessitate the initial set up of n^2 PSKs to uniquely identify each peer. Otherwise, i.e., where 3 or more peers share the same PSK, peers would be unable to distinguish one communication partner from the other. Moreover, dynamically changing the network components becomes inefficient, e.g., if a new peer is added to the network, fresh PSKs have to be distributed to all other peers off-channel.

Password-based authenticated key exchanges [BPR00,BMP00] are more interesting in a multi-client, single-sever scenario where each client is uniquely identified using a (low-entropy) password. Similar to the PSK approach, the password is an intrinsic part of the exchange and cannot be completed without knowledge of the specific password. As the scenario we are considering is not a multi-client single-server scenario, we will omit further discussions of this type of key exchange.

Finally, with certificate-based protocols, peers have long-term public keys (typically of a digital signature scheme) whereas certificate authorities ensure the

authenticity of these keys and establish a chain of trust. During a protocol run, peers are then required to sign certain messages to authenticate the exchange. A prominent example of such a protocol is SIGMA [Kra03] which serves as a prototype for the key exchange deployed in IPsec, for example. Recently, due to the bandwidth requirements of post-quantum secure signature schemes, variants with long-term KEM keys such as KEMTLS [SSW20] are also gaining interest.

While PSK key exchanges can be implemented solely from symmetric-key primitives, managing the required keys is a complex task. As no key material is available during system setup, those keys need to be securely exchanged via trusted couriers, installed on devices in the fab, or other methods are required to allow the keys be installed without relying on a yet unsecured communication channel. This task becomes more complex as the network grows and infeasible if parties have no trivial way to securely exchange the PSK.

Quantum-Safe Authenticated Key Exchanges. End-to-end post-quantum AKE protocols have already been (experimentally) studied, e.g., most prominently in the area of TLS [BCNS15,Lan16,KSL+19,PST20,SSW20]. Moreover, standardization efforts towards post-quantum (hybrid) key exchanges is already in progress while NIST is expected to publish the first standards on post-quantum key-exchange mechanisms and digital signatures soon.[3] For most practical use-cases that require security against cryptographically relevant quantum computers, the post-quantum cryptography (PQC) paradigm seems to be a strong fit, although some techniques are rather recent and severe attacks are happening [Beu22].

For highly secure use-cases, quantum-key distribution (QKD) [ABB+14,MNR+20] is gaining quite some attention recently and some big companies even expect a market growth of 12 billion USD in 10 years.[4] Moreover, there is the European initiative for a quantum communication infrastructure named EuroQCI.[5] The benefit of a QKD system is that it guarantees information-theoretic security compared to computational security of post-quantum primitives. However, QKD comes with significant limitations such as range and costly hardware.

Moreover, QKD requires PSK-based authentication. Noteworthy, the PSKs for the individual QKD links are not enough to establish authenticity for the full path through the network as they only ensure authenticity for one link. Without end-to-end authenticity, all nodes in between are turned into so-called *trusted nodes* [MNR+20]. With trusted nodes, however, deployment in large-scale networks may become even more complex.[6] Hence, practical end-to-end

[3] `https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-05.html`, `https://csrc.nist.gov/projects/post-quantum-cryptography`

[4] `https://www.reuters.com/article/us-toshiba-cyber-idUSKBN2730KW`

[5] `https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci`

[6] Interestingly, while some approaches even backed by patents (`https://www.ipo.gov.uk/p-ipsum/Case/PublicationNumber/GB2590064`) claim to provide long-range

authenticity guarantees for the to-be-anticipated QKD networks are still under investigation.

Since both, the PQC and QKD paradigms, have benefits and downsides, and following the approach "Don't put all eggs in your basket," we are interested in how to achieve end-to-end authentication and confidentiality for key exchanges with the best possible security guarantees against future threats. One promising approach is using hybrid techniques.

Hybrid Authenticated Key Exchanges. Hybrid AKE (HAKE) approaches are getting more popular nowadays and were presented in several flavors to incorporate classical, PQC, or QKD components, e.g., [MSU13,BFG19,BBF+19,DHP20]. The main benefit is redundancy, i.e., if some of the components fail, the primitive still yields a confidential and authenticated channel. Moreover, HAKE provides an approach towards the transition of non-quantum secure networks to quantum-secure ones.

Particularly interesting is the recently proposed HAKE framework with its instantiation dubbed Muckle due to Dowling, Brandt Hansen, and Paterson [DHP20]. Muckle combines secret keys obtained from a QKD network with session keys obtained from a classical and post-quantum secure key encapsulation mechanisms (KEMs). The combination of the keys is performed using a sequence of pseudo-random function evaluations.

Importantly, Muckle even achieves very desirable advanced security guarantees. Namely, forward and post-compromise security (which are de-facto standard features in the literature on AKE mechanisms nowadays). The first guarantees that prior session keys cannot be retrieved (even if the current session key leaks) while the latter guarantees that future sessions are safe again (once the adversary does not compromise the system anymore). For example, even if the classic KEM fails (e.g., in the event of a cryptographically relevant quantum computer), at least old session keys stay safe due to the PQC and QKD guarantees. Moreover, if additionally the post-quantum KEM fails (e.g., in the event of a fundamental break of the underlying assumption), an adversary cannot retrieve old sessions keys due to the QKD guarantees. Conversely, if the QKD component fails (e.g., due to side-channel attacks [BBC+21]), the security guarantees of the PQC components prevent an adversary from retrieving old session keys.

Such a forward-security guarantee is particularly important for future threats. Namely, solely using such an AKE with a single primitive, e.g., post-quantum KEMs (without QKD), puts *all* sessions immediately in danger, e.g., in the case that some post-quantum assumption does not hold. Having "store-now-decrypt-later" attacks in mind, such a risk should be mitigated in many scenarios and hybridization hedges against such threats. On the other side, even if the QKD guarantees are void (e.g., by an attacker compromising a trusted node or failing hardware), the post-quantum KEM guarantees end-to-end security for all sessions.

QKD networks without trusted nodes (i.e., establishing a secure channel between any two nodes), a recent work [HAD+22] demonstrates that such claim cannot be met.

The Muckle authentication, however, solely relies on the presence of pre-shared keys. Consequently, Muckle inherits the key management problem of PSKs in large-scale networks discussed above. In this work, we present an extension of the HAKE framework in [DHP20] and present an amplification of their Muckle scheme with end-to-end authenticity and better efficiency while no sacrifices on the security guarantees have to be made.

1.1 Contribution

Our contribution can be summarized as follows:

- We extend Muckle with certificate-based authentication mechanisms dubbed Muckle+ while preserving quantum (i.e., post-quantum) security. While replacing pre-shared keys with digital signatures is rather straightforward in general, this turned out to be surprisingly non-trivial when applied to HAKE frameworks (resulting in a significant model change with adapted proof techniques). The benefits are that we avoid the usage of PSKs (with its inherent quadratic blow-up to achieve end-to-end authenticity) which results in more efficient end-to-end HAKE instantiation than previously known.
  While gaining significant efficiency and flexibility with our approach compared to Muckle, to retrieve the same security guarantees, we need that the QKD keys are distributed via multi-path techniques.
- We implement the Muckle+ protocol and validated its functionality using a small QKD network in the field. To the best of our knowledge, such a proof-of-concept experiment for HAKEs is the first one with QKD hardware. Thereby, we can demonstrate the added authenticity guarantees that ensure an end-to-end secure connection between the initiator and responder.

More on Extending Muckle. The Muckle protocol uses a hybrid approach combining classical, post-quantum, and QKD keys through the use of a key derivation function. Muckle requires a classical and post-quantum KEM as well as data from a QKD channel to create the final shared secret. Additionally, the protocol relies on a secure pseudorandom function and a MAC. The latter is used in combination with a QKD pre-shared key to ensure the authenticity of the key exchange. To avoid such pre-shared keys for authentication, we carefully extend Muckle to allow certificate-based authentication. Technically, we use digital signatures as a building block instead of PSKs for authentication.

However, replacing PSKs with digital signatures in HAKE is not straightforward. Using PSKs yield an interesting cryptographic feature, namely, it guarantees that a sender and a receiver share a common secret key for end-to-end authentication (leaving the quadratic blow-up in that case on the side for a moment). Now, if digital signatures are used, we cannot build on such guarantee anymore (as we are in the public-key setting).

The key observation in the HAKE realm is that in the latter case, we either require a post-quantum KEM or we need multi-path approaches for the QKD part to guarantee end-to-end authenticity again. As we want to allow that the

post-quantum KEM components to fail (as in Muckle), we need that the QKD keys are distributed using a multi-path approach (essentially, by distributing key components via mutually disjoint paths from the initiator to the responder such that no individual trusted node knows all of the key material depending on some bound of colluding nodes).

Through this alteration, we achieve the desired security properties, i.e., we are able to endure all security claims from original Muckle (in particular, forward and post-compromise security) while avoiding PSKs, which we show by formally proving our variant Muckle+ secure in the HAKE framework. Moreover, our instantiation allows for an efficient approach to achieve end-to-end security which we justify via an implementation.

Implementing Muckle+. The implementation of Muckle+ to demonstrate its efficacy follows the typical structure of both a QKD security application in the sense of the ETSI QKD GS standard documents (and in particular, ETSI QKD GS 014 [ETS19]) and an authenticated key exchange using application well-understood from their use on the modern web. Thereby, the initiator of the connection obtains a key ID and the corresponding key material from a QKD device and transmits the key ID as part of the initial authenticated key exchange message to the receiver.

By providing an interface the applications that follow the structure of deployed authenticated key exchanges, we expect to reduce the required effort to integrate the use of QKD keys into applications that are already using TLS [Res18], QUIC [LRW+17], or similar protocols. Except for configuring the connection to the local QKD end-point, no further configuration will be necessary to establish secure channels with any service deployed on the QKD network.

1.2 Related Work

Authenticated key exchanges have a long history and are still a very active area of research as they represent the core component of any protocol for secure communication. Notably, Krawczyk's Sign-and-MAC (SIGMA) protocols [Kra03] serve as a template for many of the protocols used in practice. The basic idea of this template is to combine an ephemeral key exchange using key encapsulation methods (KEMs) to exchange a fresh shared secret, a signature scheme for authentication of the communication parties as well as message authentication code (MAC) to authenticate the shared secret. Keys are derived using a pseudorandom function (PRF). One execution then runs roughly as follows: the initiator produces a new ephemeral KEM key and sends the public key to the responder. The responder then performs the key encapsulation using the received public key, signs the produced ciphertext together with the first message to authenticate itself, and derives a shared secret to authenticate the session using the MAC. Ciphertext, authentication tag and signature are sent to the initiator. The initiator then decapsulates the shared secret key, verifies the received signature as well as the authentication tag. In a mutual authentication setting,

the initiator also authenticates itself using the signature scheme, but the session is also always authenticated by the initiator using the MAC. This information is sent to the responder for verification. Afterwards, the two parties share an authenticated and fresh secret key.

While SIGMA was originally proposed using Diffie-Hellman for the ephemeral key exchange, presenting it in terms of KEMs allows us to consider it in a post-quantum setting as we then can instantiate all build blocks using post-quantum secure schemes. It can also be extended with responder or initiator privacy features [Zha16,SSL20,RSW21], whereas the latter be observed in practice as part of the TLS handshake. With the migration towards post-quantum secure protocols, work on adapting and improving key exchanges protocols based on the performance and bandwidth characteristics of post-quantum secure key encapsulation mechanisms and digital signature schemes has commenced [BCNS15,SM16,HKSU20,HNS+21], though. Notably, Schwabe, Stebila and Wiggers proposed KEMTLS [SSW20], a unilaterally authenticated key exchange protocol where authentication of the responders is performed using a long-term KEM key. The basic idea is, that after establishing an ephemeral key, the initiator encapsulates a secret with respect to the responder's long-term KEM key. The responder can only produce the authentication tags for session authentication if it can decapsulate using its long term KEM key. Thereby, the responder is implicitly authenticated via its knowledge of the corresponding private key.

In the area of QKD networks, proposals exist to address the trusted-node problem with secret-sharing based multipath protocols, e.g. [RK11,RKJ+21], to exchange the secret key. In a similar vein, multipath authentication protocols have been proposed too, whereas those are built on the assumption that an adversary is unable to compromise multiple nodes in the network. When considering network topologies with many routes available for connecting any two nodes, it is therefore possible to split sensitive information into parts (e.g., via secret sharing) and to send the shares via multiple paths instead of one.

For example, Rass and Schartner [RS10] introduced a MAC-based multipath authentication protocol specifically for the application in quantum networks. In the scenario they consider, two nodes wanting to communicate in a QKD network may not necessarily establish pre-shared keys. There are however shared QKD secrets between every node and each of its immediate neighbors. The protocol uses those secrets in combination with a multipath approach to share an authenticated message between the nodes and relies on the assumptions that (a) keys created by two adjacent nodes via the QKD channel are secure, and (b) every node shares a secret key with its neighboring nodes. While the protocol is secure against $k < n$ compromised paths with executed with n disjoint paths, it does not fit into the typical notion of an authenticated key exchange and it lacks end-to-end authenticity.

Finally, secure multipath key exchange (SMKEX) [CCG+18] utilizes two disjoint paths to facilitate authentication and key exchange. The protocol is based on a typical key exchange, but in addition the second channel is used to send a random nonce that is authenticated using the secret key exchanged via the first

channel. SMKEX therefore ensures unilateral authenticity and computational security against an active adversary as long as only one path is compromised.

2 Preliminaries

In this section we briefly recall notions related to (hybrid) authenticated key exchanges.

2.1 Pseudo-random functions

Definition 1 (PRF). *Let $\mathcal{F}\colon \mathcal{S} \times D \to \mathsf{R}$ be a family of functions and let Γ be the set of all functions $D \to \mathsf{R}$. For a PPT distinguisher $\mathcal{D}$ we define the advantage function as*

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{D},\mathcal{F}}(\kappa) = \left|\Pr\left[s \xleftarrow{R} \mathcal{S} : \mathcal{D}^{\mathcal{F}(s,\cdot)}(1^\kappa) = 1\right] - \Pr\left[f \xleftarrow{R} \Gamma : \mathcal{D}^{f(\cdot)}(1^\kappa) = 1\right]\right|.$$

$\mathcal{F}$ is a pseudorandom function (family) if it is efficiently computable and for all PPT distinguishers $\mathcal{D}$ there exists a negligible function $\varepsilon(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{D},\mathcal{F}}(\kappa) \leq \varepsilon(\kappa).$$

A PRF $\mathcal{F}$ is a dual PRF [BL15], if $\mathcal{G} : D \times \mathcal{S} \to \mathsf{R}$ defined as $\mathcal{G}(d, s) = \mathcal{F}(s, d)$ is also a PRF.

2.2 Digital Signatures and Message Authentication Codes

We recall the notion of message authentication codes and digital signature schemes and the standard unforgeability notions below.

Definition 2 (Message Authentication Codes). *A message authentication code* MAC *is a triple* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *of PPT algorithms, which are defined as:*

$\mathsf{KGen}(1^\kappa)$: *This algorithm takes a security parameter κ as input and outputs a secret key* sk.

$\mathsf{Auth}(\mathsf{sk}, m)$: *This algorithm takes a secret key $\mathsf{sk} \in \mathcal{K}$ and a $m \in \mathcal{M}$ and outputs an authentication tag τ.*

$\mathsf{Ver}(\mathsf{sk}, m, \tau)$: *This algorithm takes a secret key* sk*, a message $m \in \mathcal{M}$ and an authentication tag τ as input and outputs a bit $b \in \{0, 1\}$.*

A MAC is correct if for all $\kappa \in \mathbb{N}$, for all $\mathsf{sk} \leftarrow \mathsf{KGen}(1^\kappa)$ and for all $m \in \mathcal{M}$, it holds that $\Pr\left[\mathsf{Ver}(\mathsf{sk}, m, \mathsf{Auth}(\mathsf{sk}, m)) = 1\right] = 1$.

Definition 3 (EUF-CMA). *For a PPT adversary $\mathcal{A}$, we define the advantage function in the sense of existential unforgeability under chosen message attacks (*EUF-CMA*) as*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{MAC}}(1^\kappa) = \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{MAC}}(1^\kappa) = 1\right],$$

where the corresponding experiment is depicted in Experiment 1. If for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{MAC}}(1^\kappa) \leq \varepsilon(\kappa)$, we say that MAC *is* EUF-CMA *secure.*

$\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{MAC}}(1^\kappa)$:
  $\mathsf{sk} \leftarrow \mathsf{KGen}(1^\kappa)$, $\mathcal{Q} \leftarrow \emptyset$
  $(m^*, \tau^*) \leftarrow \mathcal{A}^{\mathsf{Auth}', \mathsf{Ver}'}()$
    where oracle $\mathsf{Auth}'(m)$:
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
      return $\mathsf{Auth}(\mathsf{sk}, m)$
    where oracle $\mathsf{Ver}'(m, \tau)$:
      return $\mathsf{Ver}(\mathsf{sk}, m, \tau)$
  return 1, if $\mathsf{Ver}(\mathsf{sk}, m^*, \tau^*) = 1 \ \wedge \ m^* \notin \mathcal{Q}$, return 0 otherwise

Experiment 1: EUF-CMA security experiment for a MAC MAC.

Definition 4 (Signature Scheme). *A signature scheme* Σ *is a triple* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KGen}(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space* $\mathcal{M}$ *(we may omit to make the message space* $\mathcal{M}$ *explicit).*

$\mathsf{Sign}(\mathsf{sk}, m)$: *This algorithm takes a secret key* sk *and a message* $m \in \mathcal{M}$ *as input and outputs a signature* σ*.*

$\mathsf{Ver}(\mathsf{pk}, m, \sigma)$: *This algorithm takes a public key* pk*, a message* $m \in \mathcal{M}$ *and a signature* σ *as input and outputs a bit* $b \in \{0, 1\}$*.*

We require a signature scheme to be correct and to provide existential unforgeability under adaptively chosen message attacks (EUF-CMA security). For correctness, we require that for all $\kappa \in \mathbb{N}$, for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$ and for all $m \in \mathcal{M}$ it holds that $\Pr[\mathsf{Ver}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1] = 1$.

Definition 5 (EUF-CMA). *For a PPT adversary* $\mathcal{A}$*, we define the advantage function in the sense of existential unforgeability under chosen message attacks (*EUF-CMA*) as*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(1^\kappa) = \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(1^\kappa) = 1\right],$$

where the corresponding experiment is depicted in Experiment 2. If for all PPT adversaries $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(1^\kappa) \leq \varepsilon(\kappa)$*, we say that* Σ *is* EUF-CMA *secure.*

2.3 Key-Encapsulation Mechanisms

Definition 6. *A key-encapsulation mechanism (KEM) scheme* KEM *with key space* $\mathcal{K}$ *consists of the three PPT algorithms* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(1^\kappa)$:
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$, $\mathcal{Q} \leftarrow \emptyset$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}}(\mathsf{pk})$
    where oracle $\mathsf{Sign}'(m)$:
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
      return $\mathsf{Sign}(\mathsf{sk}, m)$
  return 1, if $\mathsf{Ver}(\mathsf{pk}, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{Q}$, return 0 otherwise

Experiment 2: EUF-CMA security experiment for a digital signature scheme Σ.

$\mathsf{KGen}(1^\kappa)$: *On input security parameter κ, outputs public and secret keys* $(\mathsf{pk}, \mathsf{sk})$.
$\mathsf{Enc}(\mathsf{pk})$: *On input* pk*, outputs a ciphertext c and key K.*
$\mathsf{Dec}(\mathsf{sk}, c)$: *On input* sk *and c, outputs K or $\{\perp\}$.*

We call a KEM correct if for all $\kappa \in \mathbb{N}$, for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\kappa)$, for all $(c, K) \leftarrow \mathsf{Enc}(\mathsf{pk})$, we have that

$$\Pr[\mathsf{Dec}(\mathsf{sk}, c) \neq K] = 0.$$

Definition 7. *For a PPT adversary $\mathcal{A}$, we define the advantage function in the sense of indistinguishability under chosen plaintext attacks (KEM-IND-CPA) and indistinguishability under chosen ciphertexts attacks (KEM-IND-CCA) as*

$$\mathsf{Adv}^{\mathsf{kem\text{-}ind\text{-}cpa}}_{\mathcal{A},\mathsf{KEM}}(1^\kappa) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{kem\text{-}ind\text{-}cpa}}_{\mathcal{A},\mathsf{KEM}}(1^\kappa) = 1\right] - \frac{1}{2} \right|, \text{ and}$$
$$\mathsf{Adv}^{\mathsf{kem\text{-}ind\text{-}cca}}_{\mathcal{A},\mathsf{KEM}}(1^\kappa) = \left| \Pr\left[\mathsf{Exp}^{\mathsf{kem\text{-}ind\text{-}cca}}_{\mathcal{A},\mathsf{KEM}}(1^\kappa) = 1\right] - \frac{1}{2} \right|$$

where the corresponding experiments are depicted in Experiment 3, respectively. If for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{kem\text{-}ind\text{-}cpa}}_{\mathcal{A},\mathsf{KEM}}(1^\kappa) \leq \varepsilon(\kappa), \text{ or} \mathsf{Adv}^{\mathsf{kem\text{-}ind\text{-}cca}}_{\mathcal{A},\mathsf{KEM}}(1^\kappa) \leq \varepsilon(\kappa)$$

we say that KEM *is KEM-IND-CPA or KEM-IND-CCA secure, respectively.*

2.4 Authenticated Key Exchange

We recall the hybrid authenticated key exchange (HAKE) security model by Dowling et al. [DHP20]. For a general treatment of authenticated key exchanges (AKE) we refer the reader to [DvOW92,KL14].

$\mathsf{Exp}^{\mathsf{kem\text{-}ind}\text{-}T}_{\mathcal{A},\mathsf{KEM}}(\kappa)$:
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$
$(c^*, K_0) \leftarrow \mathsf{Enc}(\mathsf{pk}), K_1 \xleftarrow{R} \mathcal{K}$
$\mathcal{Q} \leftarrow \emptyset$, $b \xleftarrow{R} \{0,1\}^\kappa$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}, c^*, K_b)$
where $\mathcal{O} = \{\mathsf{Dec}'\}$ if $T = \mathsf{cca}$ with oracle $\mathsf{Dec}'(c)$:
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{c\}$
return $\mathsf{Dec}(\mathsf{sk}, c)$
return 1, if $b = b^* \ \wedge \ c^* \notin \mathcal{Q}$, return otherwise 0

Experiment 3: KEM-IND-T security experiments for KEM with $T \in \{\mathsf{cpa}, \mathsf{cca}\}$.

Execution Environment. We consider a set of n_P parties $P_1, \ldots, P_{n_P}$ which are able to run up to n_S sessions of a key exchange protocol between them, where each session may consist of n_T different stages of the protocol. Each party P_i has access to its long-term key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and to the public keys of all other parties. Each session is described by a set of session parameters:

- $\rho \in \{\mathsf{init}, \mathsf{resp}\}$: The role (initiator or responder) of the party during the current session
- $pid \in n_P$: The communication partner of the current session
- $stid \in n_T$: The current stage of the session
- $\alpha \in \{\mathsf{active}, \mathsf{accept}, \mathsf{reject}, \perp\}$: The status of the session. Initialized with $\perp$.
- $m_i[stid], i \in \{s, r\}$: All messages sent ($i = s$) or received ($i = r$) by a session up to the stage *stid*. Initialized with $\perp$.
- $k[stid]$: All session keys created up to stage *stid*. Initialized with $\perp$.
- $exk[stid], x \in \{q, c, s\}$: All ephemeral post-quantum(q), classical(c) or symmetric(s) secret keys created up to stage *stid*. Initialized with $\perp$.
- $pss[stid]$: The per session secret state ($\mathsf{SecState}$) that is created during the stage *stid* for the use in the next stage.
- $st[stid]$: Storage for other states used by the session in each stage.

We describe the protocol as a set of algorithms $(f, \mathsf{KGen}XY, \mathsf{KGen}ZS)$:

- $f(\lambda, \mathsf{pk}_i, \mathsf{sk}_i, pskid_i, psk_i, \pi, m) \rightarrow (m', \pi')$: a probabilistic algorithm that represents an honest execution of the protocol. It takes a security parameter λ, the long-term keys $\mathsf{pk}_i, \mathsf{sk}_i$, the session parameters π representing the current state of the session and a message m and outputs the updated session state π' and a response m'.
- $\mathsf{KGen}XY(\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$: a probabilistic asymmetric key generation algorithm that takes a security parameter λ and creates a public-key, secret-key pair $(\mathsf{pk}, \mathsf{sk})$. $X \in \{E, L\}$ determines whether the created key is an ephemeral (E) or long-term (L) secret. $Y \in \{Q, C\}$ determines whether the key is classical (C) or post-quantum(Q).

– $\mathsf{KGen}ZS(\lambda) \rightarrow (psk, pskid)$: a probabilistic symmetric key generation algorithm that takes a security parameter λ and outputs symmetric keying material (psk). $Z \in \{E, L\}$ determines whether the created key is an ephemeral (E) or long-term (L) secret.

For each party $P_1, \ldots, P_{n_P}$, classical as well as post-quantum long-term keys are created using the corresponding $\mathsf{KGen}XY$ algorithms. The challenger then randomly chooses a bit $b \in 0, 1$ that will determine the key returned by the Test query. From this point on the adversary may interact with the challenger using the queries defined in the next section. At some point during the execution of the protocol, the adversary $\mathcal{A}$ may issue the Test query and present a guess for the value of b. If $\mathcal{A}$ guesses correctly and the session satisfies the cleanness predicate, the adversary wins the key indistinguishability Game.

Adversarial Interaction. The HAKE framework defines a range of queries that allow the attacker to interact with the communication

– $\mathsf{Create}(i, j, role) \rightarrow \{(s), \perp\}$: Initializes a new session between party P_i with role 'role' and the partner P_j. If the session already exists the query returns $\perp$ otherwise the session (s) is returned.
– $\mathsf{Send}(i, s, m) \rightarrow \{m', \perp\}$: Enables $\mathcal{A}$ to send messages to sessions and receive the response m' by running f for the session π_i^s. Returns $\perp$ if the session is not active.
– $\mathsf{Reveal}(i, s, t)$: Provides $\mathcal{A}$ with the session keys corresponding to a session π_i^s if the session is in the accepted state. Otherwise, $\perp$ is returned.
– $\mathsf{Test}(i, s, t) \rightarrow \{k_b, \perp\}$: Provides $\mathcal{A}$ with the real (if b=1) or random (b=0) session key for the key indistinguishably game.
– $\mathsf{Corrupt}XY(i) \rightarrow \{key, \perp\}$: Provides $\mathcal{A}$ with the long-term $XY \in \{\mathsf{SK}, \mathsf{QK}, \mathsf{CK}\}$ keys for P_i. If the key has been corrupted previously, $\perp$ is returned. Specifically:
  • $\mathsf{CorruptSK}$: Reveals the pre-shared key
  • $\mathsf{CorruptQK}$: Reveals the post-quantum long-term key
  • $\mathsf{CorruptCK}$: Reveals the classical long-term key
– $\mathsf{Compromise}XY(i, s, t) \rightarrow \{key, \perp\}$: Provides $\mathcal{A}$ with the ephemeral $XY \in \{\mathsf{QK}, \mathsf{CK}, \mathsf{SK}, \mathsf{SS}\}$ keys created during the session π_i^s prior to stage t. If the ephemeral key has already been compromised, $\perp$ is returned. Specifically:
  • $\mathsf{CompromiseQK}$: Reveals the ephemeral post-quantum key
  • $\mathsf{CompromiseCK}$: Reveals the ephemeral classical key
  • $\mathsf{CompromiseSK}$: Reveals the ephemeral quantum key
  • $\mathsf{CompromiseSS}$: Reveals the ephemeral per session state ($\mathsf{SecState}$)

Matching Sessions. Furthermore, we recall the definitions of matching sessions [LKZC07] and origin sessions [CF12] which covers that the two parties involved in a session have the same view of their conversation.

Definition 8 (Matching sessions). *We consider two sessions π_i^s and π_j^r in stage t to be matching if all messages sent by the former session $\pi_i^s.m_s[t]$ match*

those received by the later $\pi_j^r.m_r[t]$ and all messages sent by the later session $\pi_j^r.m_s[t]$ are received by the former $\pi_i^s.m_r[t]$.

π_i^s considered to be prefix-matching with π_j^r if $\pi_i^s.m_s[t] = \pi_j^r.m_r[t]'$ where $\pi_j^r.m_r[t]$ is truncated to the length of $\pi_i^s.m_s[t]$ resulting in $\pi_j^r.m_r[t]'$.

Definition 9 (Origin sessions). *We consider a session π_i^s to have an origin session with π_j^r if π_i^s matches π_j^r or if π_i^s prefix-matches π_j^r.*

Security. Dowling et al. define key indistinguishability with respect to a predicate clean. However, their predicate is specific to Muckle and hence we therefore only give the security notion and postpone the discussion of the predicate to Section 3.3.

Definition 10. *Let Π be a key-exchange protocol and $n_P, n_S, n_T \in \mathbb{N}$. For a predicate clean and an adversary $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the HAKE key-indistinguishability game as*

$$\mathsf{Adv}^{HAKE,\mathsf{clean},\mathcal{A}}_{\Pi,n_P,n_S,n_T}(\kappa) = \left|\Pr\left[\mathsf{Exp}^{HAKE,\mathsf{clean},\mathcal{A}}_{\Pi,n_P,n_S,n_T}(\kappa) = 1\right]\right|.$$

We say that Π is HAKE-security if $\mathsf{Adv}^{HAKE,\mathsf{clean},\mathcal{A}}_{\Pi,n_P,n_S,n_T}(\kappa)$ is negligible in the security parameter κ for all $\mathcal{A}$.

If security also holds against any quantum algorithm $\mathcal{A}$, then we call Π post-quantum secure.

3 Extending Muckle with Signature-Based Authentication

In this section, we recap Muckle [DHP20] and present our novel variant Muckle+.

3.1 Muckle

The Muckle protocol uses a hybrid approach combining classical, post-quantum, and QKD keys through the use of a key derivation function. More concretely, Muckle requires classical and post-quantum key encapsulation mechanisms (KEMs) as well as data from a QKD channel (key k_q) to create the final shared secret between communication partners.

Muckle is a multi-stage protocol. While a Muckle instance is active between two parties, a single stage is run repeatedly, creating a pair of session keys during each execution. The communication that occurs during one stage of the protocol is detailed in Figure 1.

The Muckle key exchange requires a symmetric pre-shared key PSK and unique party identifiers (implicit in ℓ_I and ℓ_R) to be distributed to the communication partners before the key exchange. The parties also have to set an initial value for the session secret state $\mathsf{SecState}$. To begin a new session, the initiator

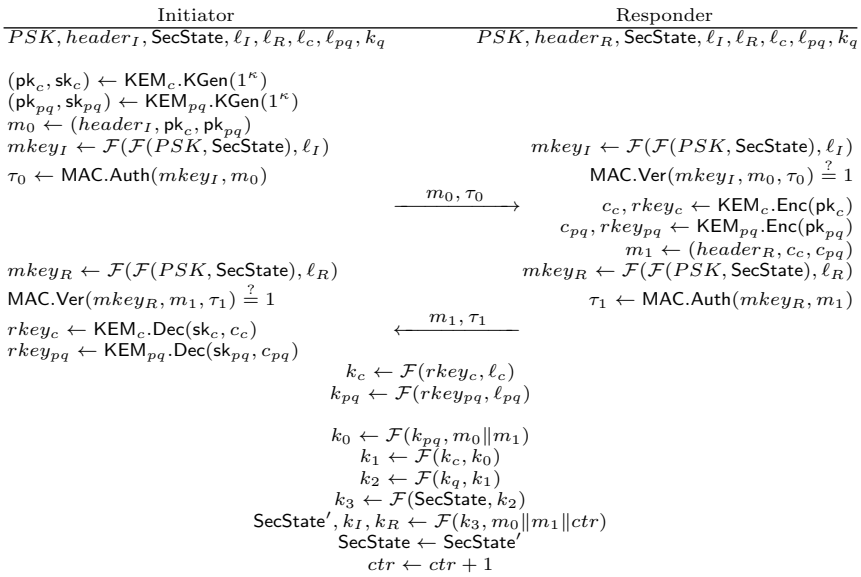

Fig. 1. One stage of the Muckle protocol [DHP20] with a classical KEM KEM_c, a post-quantum KEM KEM_{pq}, a MAC MAC, and a PRF $\mathcal{F}$ whereas k_q represents the key provided by the QKD link which is provided out-of-band.

uses the classical KEM KEM_c and post-quantum KEM KEM_{pq} to create a classical key pair $(\mathsf{pk}_c, \mathsf{sk}_c)$ and a post-quantum key pair $(\mathsf{pk}_{pq}, \mathsf{sk}_{pq})$, respectively. Both public keys are then combined with a header containing meta-data into the message m_0. The PRF $\mathcal{F}$ is applied over PSK and $\mathsf{SecState}$ to create a unique value for the current session, which is then used as an input in another round of the PRF with the value ℓ_I resulting in the message key $mkey_I$. The key $mkey_I$ is used as the MAC key to create a tag τ_0 for the message m_0. The message m_0 and the tag τ_0 are then sent to the responder.

Receiving the transmission, the responder will check the authenticity of the message m_0 by verifying the tag τ_0 with its $mkey_I$ (where $mkey_I$ is derived via PSK, shared $\mathsf{SecState}$, and ℓ_I). If the verification succeeds, the responder can now use the encapsulation functions of the KEMs to create the keys $rkey_c$ and $rkey_{pq}$ as well as the ciphertexts c_c and c_{pq}, respectively. The responder proceeds to create a message m_1 and a tag τ_1 analogously to the initiator's MAC procedure, but using the ciphertexts instead of the public keys and the responder value $header_R$.

m_1 and τ_1 are then transmitted to the initiator, who can use them in the KEM decapsulation function to get the keys $rkey_c$ and $rkey_{pq}$ (after successful verification of m_1). From this point on, the initiator and responder share the same information and proceed with the same steps.

Table 1. Comparison of the protocols in terms of provided security guarantees: KC (key confirmation), PFS (perfect forward secrecy), PCS (post-compromise security).

	protocol	authentication		KC	PFS	PCS
multipath	MAC-based [RS10]	?	initiator			
	SMKEX [CCG+18]	?	responder		*	*
post-quantum	SIGMA [Kra03]	explicit	mutual	✓	✓	
	KEMTLS [SSW20]	explicit[1]	responder/mutual	✓[2]	✓	?[3]
	Muckle [DHP20]	explicit	mutual	✓	✓	✓[4]

[*] not applicable (no long term secret)
[1] implicit for client during mutual authentication
[2] only for responder authentication
[3] PCS is not explicitly shown
[4] under the conditions discussed in Section 3.1

First the both keys are entered into the PRF $\mathcal{F}$ together with labels ℓ_c and ℓ_{pq} to create the further keys. Then the key schedule starts combining all the keys into a final shared secret k_I, k_R and setting a new session state as well as incrementing the session counter.

Muckle offers mutual authentication, perfect forward secrecy (PFS) and post-compromise security (PCS). Post-compromise security is guaranteed under the condition that at least one previous stage has been completed without the attacker compromising all the ephemeral (classical, quantum, post-quantum and session secret) secrets, and that the attacker has been only acting passively since then.

3.2 Extending Muckle with Signature-Based Authentication

In Table 1, we compare the security properties of the protocols we have discussed in Section 1.2 and Muckle. From this comparison, we can conclude that Muckle offers the most features and is therefore a suitable candidate for realizing end-to-end secure hybrid authenticated key exchanges. However, the protocol relies on PSKs for end-to-end authentication. As the other components including the QKD layer do not provide end-to-end authentication (cf. Section 3.1), we extend Muckle to also offer mutual signature-based authentication. Through this alteration, we preserve the desirable security whilst avoiding the issues associated with PSKs. We will from now on refer to this new protocol as Muckle+.

Like Muckle, Muckle+ is a multi-stage protocol. One such stage is detailed in Figure 2. The basic structure of Muckle+ is very similar to the original Muckle protocol. Up to the computation of the final chaining key, the PSK-based authentication is replaced with signature-based authentication and the addition of two random nonces n_I and n_R to avoid issues with the reuse of signatures. We note that the modifications essentially correspond to changing to a SIGMA-style

key exchange with multiple KEMs and an additional PSK that is provided by the QKD link.

Fig. 2. One stage of the Muckle+ protocol. Messages $\boldsymbol{m}$: $\{m_1, \ldots\}_k$ denote that $m_1, \ldots$ is encrypted with an authenticated encryption scheme using the secret key k. The various contexts and labels are given in Tables 2 and 3.

3.3 Security of Muckle+

Similar to Muckle, Muckle+ achieves the same security properties including post-compromise security (PCS) and perfect forward secrecy (PFS). In this section,

Table 2. Values for the contexts used in the Muckle+ key schedule. The context inputs follow the choices in the TLS 1.3 handshake [DFGS21].

Label	Context Input	Label	Context Input
H_ε	""	H_0	$H(\text{""})$
H_1	$H(m_1 \| m_2)$	H_2	$H(m_1 \| \ldots \| m_3)$
H_3	$H(m_1 \| \ldots \| m_4)$		

Table 3. Values for the labels used in the Muckle+ key schedule for domain separation. Some of these labels are directly based on the corresponding labels in the TLS 1.3 handshake [DFGS21]. The concrete value of these labels is unimportant as long as they are unique.

Label	Label Input	Label	Label Input
ℓ_0	"derive k c"	ℓ_1	"derive k pq"
ℓ_2	"first ck"	ℓ_3	"second ck"
ℓ_4	"third ck"	ℓ_5	"fourth ck"
ℓ_6	"derived"	ℓ_7	"c hs traffic"
ℓ_8	"s hs traffic"	ℓ_9	"finished"
ℓ_{10}	"c ap traffic"	ℓ_{11}	"s ap traffic"
ℓ_{12}	"secstate"	ℓ_{13}	"TLS 1.3, server CertificateVerify"
ℓ_{14}	"TLS 1.3, client CertificateVerify"		

we formally proof this claim. The presented security analysis of the Muckle+ protocol is based on the HAKE framework as introduced by Dowling et al. [DHP20]. We will use the definitions and notations use in the HAKE framework in this analysis unless stated otherwise.

An adversary $\mathcal{A}$ has access all queries defined in the HAKE framework. As no pre-shared key exists in the Muckle+ protocol, the query $\mathsf{CorruptSK}$ will return $\perp$ if called. As multiple sessions keys are created in the new protocol, we specify that the key to be guessed during the Test query is the master secret MS.

We define a new cleanness predicate $\mathsf{clean}_{\text{Muckle+}}$ for our protocol that captures the same goals – post-compromise security and perfect forward secrecy – but adapt it to match our setting. As our protocol does not require a long term PSK, we can omit handling compromise of the PSK in our predicate. We however have to take care of long-term signature keys instead. Hence, we consider their compromise in $\mathsf{clean}_{\text{Muckle+}}$ as well. Overall, the goal of the cleanness predicate is to handle the compromise of as many combinations as possible as long as one set of keys – the post-quantum secure keys or the keys obtained from the QKD link – stay secure.

More formally, we define the cleanness of a session as follows: A session π_i^s in stage t is considered clean under the predicate $\mathsf{clean}_{\text{Muckle+}}$ if:

- $\mathsf{Reveal}(i, s, t)$ has not been issued for session π_i^s.
- $\mathsf{Reveal}(j, r, t)$ has not been issued for all sessions π_j^r matching π_i^s at stage t.

– If π_i^s has a matching session π_j^r, at least one of the following conditions has been met:
  - No $\mathsf{CompromiseQK}(i, s, t)$ or $\mathsf{CompromiseQK}(j, r, t)$ have been issued.
  - No $\mathsf{CompromiseSK}(i, s, t)$ or $\mathsf{CompromiseSK}(j, r, t)$ have been issued.
  - No $\mathsf{CompromiseQK}(i, s, t')$ or $\mathsf{CompromiseQK}(j, r, t)$ have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No $\mathsf{CompromiseSS}(i, s, u)$ or $\mathsf{CompromiseSS}(j, r, u)$ have been issued.
  - No $\mathsf{CompromiseSK}(i, s, t')$ or $\mathsf{CompromiseSK}(j, r, t')$ have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No $\mathsf{CompromiseSS}(i, s, u)$ or $\mathsf{CompromiseSS}(j, r, u)$ have been issued.
– If there exists no $(j, r, t) \in [n_P] \times [n_S] \times [n_T]$ such that π_j^r is an origin session of π_i^s in stage t, then either $\mathsf{CompromiseSK}(i, j, t)$ and $\mathsf{CompromiseSK}(j, i, t)$ or $\mathsf{CorruptQK}(i)$ and $\mathsf{CorruptQK}(j)$ have not been issues before $\pi_i^s.\alpha[t] \leftarrow \mathsf{accept}$. If there exists $(j, r, t) \in [n_P] \times [n_S] \times [n_T]$ such that π_j^r is an origin session of π_i^s in stage t, then either $\mathsf{CompromiseSK}(i, j, t)$ and $\mathsf{CompromiseSK}(j, i, t)$ or $\mathsf{CorruptQK}(i)$ and $\mathsf{CorruptQK}(j)$ have not been issued before $\pi_i^r.\alpha[t] \leftarrow \mathsf{accept}$.

We note that similar to $\mathsf{clean}_{\mathrm{Muckle}}$, we can define classical and quantum variants of the predicate to also reflect compromise of the classical keys. In that case, $\mathsf{clean}_{\mathrm{cMuckle+}}$ is extended to include the following two conditions for matching sessions π_i^s and π_j^r:

– No $\mathsf{CompromiseCK}(i, s, t)$ or $\mathsf{CompromiseCK}(j, r, t)$ have been issued.
– No $\mathsf{CompromiseCK}(i, s, t')$ or $\mathsf{CompromiseCK}(j, r, t')$ have been issued with π_i^s matching π_j^r in stages u where $t' \leq u \leq t$. No $\mathsf{CompromiseSS}(i, s, u)$ or $\mathsf{CompromiseSS}(j, r, u)$ have been issued.

We will now proof that the proposed protocol is secure with the cleanness predicate $\mathsf{clean}_{\mathrm{Muckle+}}$. In order to do so, we analyze the five cases corresponding to the conditions that are necessary to fulfil the $\mathsf{clean}_{\mathrm{Muckle+}}$ predicate.

Theorem 1. *The Muckle+ key exchange protocol is HAKE-secure with cleanness predicate* $\mathsf{clean}_{Muckle+}$ *assuming that the PRF* $\mathcal{F}$ *is a dual PRF, the MAC* MAC *is* EUF-CMA *secure, the KEMs* KEM_c *and* KEM_{pq} *are* IND-CPA *secure and the signature scheme* Σ *is* EUF-CMA *secure. If the security of* $\mathcal{F}$, MAC, KEM_{pq} *and* Σ *or of QKD hold against a quantum adversary, then so does the security of Muckle+.*

Proof. We divide the proof into different cases where the query $\mathsf{Test}(i, s, t)$ has been issued and prove them separately:

1. The session π_i^s (where $\pi_i^s.\rho = \mathsf{init}$) has no origin session in stage t.
2. The session π_i^s (where $\pi_i^s.\rho = \mathsf{resp}$) has no origin session in stage t.
3. The session π_i^s in stage t has a matching session.

Similar to the proof of Muckle, we show the first and the third case. The second case follows analogously to the first case.

Case 1: Test init *session without origin session.* In case 1 we show, that $\mathcal{A}$ has negligible chance of getting a session to reach the accept state if a CorruptQK or a CompromiseSK query has been issued. If the session does not reach the accept stage, the Test query will always return $\perp$, preventing $\mathcal{A}$ from winning the indistinguishability game. First we consider the case that no CorruptQK query has been issue.

Game 0: Standard HAKE-Game

$$\mathsf{Adv}^{HAKE,\mathsf{clean}_{\text{Muckle+}},\mathcal{A},C_1}_{\text{Muckle+},n_P,n_S,n_T}(\kappa) = \Pr[S_0]$$

Game 1: In Game 1, the parameters (i, s, t) for a session and its matching session (j, r, t) are guessed. If a $\mathsf{Test}(i', s', t)$ query is issued for any session $\pi^{s'}_{i'}$ that is not the test session π^s_i the game aborts.

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \Pr[S_1]$$

Game 2: Game 2 aborts, if the test session π^s_i ever reaches the status reject. As the Test query will always return $\perp$ is the session reaches this status, the advantage gained by $\mathcal{A}$ is 0.

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \Pr[S_2]$$

Game 3: Game 3 aborts, if the session reaches the status accept.

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \Pr[S_3]$$

We now bound the probability of $\mathcal{A}$ reaching the abort event. Assuming that the session reaches the status accept, we construct an EUF-CMA adversary against Σ. The challenge pk is used as the party's public key. For all other sessions, the signing oracle is used to produce the corresponding signatures. Now, if the test session reaches accept stage, we output the signature σ_I as forgery on the message $\ell_{14} \| H_3$. The signature verifies since accept stage was reached and has not been queried to the signing oracle (except for collisions of the hash function H). Hence, we obtain:

$$\Pr[S_0] \leq n_P^2 n_S n_T \cdot \left(\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\kappa)\right)$$

The case that no CompromiseSK query has been issued before reaching the accept stage, follows analogously to [DHP20, Theorem 1, Case 1] and is not repeated here.

Case 3: Test session with matching session. We will show that any adversary $\mathcal{A}$ has a negligible chance of winning the key-indistinguishability game using a sequence of games for each of the four cases. We denote with S_i the event of the adversary winning game i. Note that the proofs are the same regardless of whether $\mathsf{KEM} \in \{\mathsf{KEM}_c, \mathsf{KEM}_{pq}\}$, whereas security against a quantum adversary can only be achieved for $\mathsf{KEM} = \mathsf{KEM}_{pq}$. We split the proof into several subcases.

Subcase 1: No $\mathsf{CompromiseQK}(i,s,t)$ *or* $\mathsf{CompromiseQK}(j,r,t)$ *have been issued.* Subcase 1 shows, that if the attacker issues a Test query to a session that is clean due to the secrecy of the ephemeral post-quantum key, he has a negligible advantage in guessing the test bit. In this scenario all ephemeral secrets except the post-quantum key as well as the long-term classical and post-quantum secrets are known to the attacker.

Game 0: Standard HAKE-Game

$$\mathsf{Adv}^{HAKE,\mathsf{clean}_{\text{Muckle+}},\mathcal{A},C_2}_{\text{Muckle+},n_P,n_S,n_T}(\kappa) = \Pr[S_0]$$

Games 1-7: Games 1 to 7 for Muckle are equivalent to the Games 1 to 7 of the proof of case 3.1 as described in [DHP20], resulting in the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + 2 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

Game 8: In Game 8 the computation of the derived handshake secret dHS is replaced by a uniformly random value. To achieve this, ℓ_6 is queried together with the context input H_0 and a PRF challenger is initialized for the computation. The output of the challenger is used to replace the dHS secret. As k_3 is uniformly random by Game 7, this is a valid replacement. To distinguish between the case where $dHS \leftarrow \mathcal{F}(k_3, \ell_6, H_0)$ or $dH \xleftarrow{R} \{0,1\}^\kappa$ the attacker would have to break the **prf** security of PRF and thus has the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + 3 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

Game 9: In Game 9 the derivation of the master secret MS is replaced by a uniformly random value. A PRF challenger is initialised and its output used to replace MS. Since dHS is already random by Game 8, this is a valid substitution. To distinguish between the case, where $MS \leftarrow \mathcal{F}(dHS, 0)$ or $M \xleftarrow{R} \{0,1\}^\kappa$, $\mathcal{A}$ would have to break the **prf** security of PRF which leafs the attacker with the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + 4 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

Game 10: In Game 10 the application traffic secrets (CATS,SATS) and the session state $\mathsf{SecState}$ are replaced by a uniformly random value. This is done by initializing a PRF challenger for each computation and querying the labels 10,11 and 12 as well as the context input H_3 and replacing the corresponding value with the output from the challenger. Since the master secret MS is already random by Game 9, this is a valid substitution. For $\mathcal{A}$ to distinguish between the case where $CATS, SATS, \mathsf{SecState} \leftarrow \mathcal{F}(MS, \ell_{\{10,11,12\}}, H_3)$ or $CATS, SATS, \mathsf{SecState} \xleftarrow{R} \{0,1\}^{\mathcal{F}}$ he would have to break the **prf** security of PRF.

At this point the application traffic secrets and the session state are shown to be uniformly random under the condition of case 2 and $\mathcal{A}$ has an advantage of

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + 5 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda) \right)$$

Subcase 2: No $\mathsf{CompromiseSK}(i,s,t)$ *or* $\mathsf{CompromiseSK}(j,r,t)$ *have been issued.* This case shows, that if the attacker issues a Test query to a session that is clean due to the secrecy of the ephemeral quantum key, he has a negligible advantage in guessing the test bit. In this scenario all ephemeral secrets except the quantum key as well as the long-term classical and post-quantum secrets are known to the attacker.

Game 0: Standard HAKE-Game

$$\mathsf{Adv}^{HAKE,\mathsf{clean}_{\text{Muckle+}},\mathcal{A},C_3}_{\text{Muckle+},n_P,n_S,n_T}(\kappa) = \Pr[S_0]$$

Games 1-3: Games 1 to 3 for Muckle are equivalent to Games 1 to 3 of the proof of case 3.2 as described in [DHP20], resulting in the following advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(\mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\kappa) + \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda) \right)$$

Games 4-6: Games 4 to 6 are equivalent to Games 8 to 10 in subcase 1, resulting in the final advantage of

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T \cdot \left(4 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\kappa) + \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda) \right)$$

Subcase 3: No $\mathsf{CompromiseQK}(i,s,t')$ *or* $\mathsf{CompromiseQK}(j,r,t')$ *have been issued with* π_i^s *matching* π_j^r *in stages* u *where* $t' \leq u \leq t$. *No* $\mathsf{CompromiseSS}(i,s,u)$ *or* $\mathsf{CompromiseSS}(j,r,u)$ *have been issued.* This case shows, that if a previous session has been completed cleanly under the predicate $\mathsf{clean}_{\text{Muckle+}}$ and $\mathcal{A}$ has not compromised the session state $\mathsf{SecState}$ since then, the attacker has a negligible advantage in guessing the test bit of the current session.

Game 0: Standard HAKE-Game

$$\mathsf{Adv}^{HAKE,\mathsf{clean}_{\text{Muckle+}},\mathcal{A},C_4}_{\text{Muckle+},n_P,n_S,n_T}(\kappa) = \Pr[S_0]$$

Game 1: In Game 1 the parameters (i,s,t) for a session and its matching session (j,r,t), as well as the stage t' are guessed. If $\mathcal{A}$ issues a $\mathsf{Test}(i',s',t)$ query for any session $\pi_{i'}^{s'}$ that is not the test session π_i^s the game aborts.

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2$$

Games 2-10: Games 2 to 10 are equivalent to Games 2 to 10 in subcase 1.

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \left(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + 5 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda) \right)$$

After Game 10 the session π_i^s has been completed cleanly in stage t'. The following Games take place in each stage u and are therefore executed not once, but u-times. To represent the worst case scenario where $\mathcal{A}$ has compromised every stage after the first one, we replace the factor u by n_T.

Game 11: In Game 11 the computation of k_3 is replaced by a uniformly random value. This is done by initializing a post-quantum PRF challenger with the value k_2 and replacing k_3 with the output. As $\mathsf{SecState}$ is uniformly random by Game 10, this is a valid substitution. To distinguish between the case of $k_3 \leftarrow \mathcal{F}(\mathsf{SecState}, k_2)$ or $k_3 \xleftarrow{R} \{0,1\}^{\mathcal{F}}$ the attacker would have to break the **prf** security of the PRF resulting in the advantage:

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + (5+n_T) \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

Games 12-14: Games 12 to 14 are equivalent to Games 8 to 10 in case 2

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + (5+4n_T) \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 3 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

Subcase 4: No $\mathsf{CompromiseSK}(i,s,t')$ *or* $\mathsf{CompromiseSK}(j,r,t')$ *have been issued with* π_i^s *matching* π_j^r *in stages* u *where* $t' \leq u \leq t$. *No* $\mathsf{CompromiseSS}(i,s,u)$ *or* $\mathsf{CompromiseSS}(j,r,u)$ *have been issued.* This case shows, that if a previous session has been completed cleanly under the predicate $\mathsf{clean}_{\text{Muckle+}}$ and $\mathcal{A}$ has not compromised the session state $\mathsf{SecState}$ since then, the attacker has a negligible advantage in guessing the test bit of the current session.

Game 0: Standard HAKE-Game

$$\mathsf{Adv}^{HAKE,\mathsf{clean}_{\text{Muckle+}},\mathcal{A},C_5}_{\text{Muckle+},n_p,n_s,n_t}(\kappa) = \Pr[S_0]$$

Game 1: In Game 1 the parameters (i,s,t) for a session and its matching session (j,r,t), as well as the stage t' are guessed. If $\mathcal{A}$ issues a Test query for any session that is not the test session π_i^s the game aborts.

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \Pr[S_1]$$

Games 2-6: Games 2 to 6 are equivalent to Games 2 to 6 in subcase 2

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \Big(4 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\kappa) + \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

After Game 6 the session π_i^s has been completed cleanly in stage t'. The following Games take place in each stage u and are therefore executed not once, but u-times. To represent the worst case scenario where $\mathcal{A}$ has compromised every stage after the first one, we replace the factor u by n_T.

Games 7-10: Games 7 to 10 are equivalent to Games 11 to 14 in subcase 3.

$$\Pr[S_0] \leq n_P^2 n_S^2 n_T^2 \cdot \Big((4+4n_T) \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\kappa) + \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)$$

Finally, we obtain the following advantage.

$$\begin{aligned}
&\mathsf{Adv}^{HAKE,\mathsf{clean}_{\text{Muckle+}},\mathcal{A}}_{\text{Muckle+},n_P,n_S,n_T}(\kappa) \leq \\
&\quad n_P^2 n_S n_T \cdot \Big(\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\kappa)\Big) + \\
&\quad n_P^2 n_S^2 n_T \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + 9 \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\lambda) + 4 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big) + \\
&\quad n_P^2 n_S^2 n_T^2 \cdot \Big(\mathsf{Adv}^{ind-cpa}_{\mathsf{KEM},\mathcal{A}}(\kappa) + (9+8n_T) \cdot \mathsf{Adv}^{prf}_{\mathcal{F},\mathcal{A}}(\kappa) + 4 \cdot \mathsf{Adv}^{dual-prf}_{\mathcal{F},\mathcal{A}}(\lambda)\Big)
\end{aligned}$$

3.4 Instantiating Muckle+

Finally, we discuss some possible choices when instantiating the primitives used in Muckle+. Especially in QKD networks providing high bandwidth communication, the sizes of ciphertexts and signatures might not be a limiting factor. For the choice of signature schemes, we can thus consider candidates that are built from hash functions such as XMSS [BDH11,HBG+18] and SPHINCS+ [BHK+19] or block ciphers such as Picnic [CDG+17,KKW18] and its variant built from AES named Banquet [BdK+21]. Considering that in high bandwidth networks, the use of these symmetric primitives is perfectly valid to reduce the consumption of QKD keys, the use of these signature schemes does not require the addition of any new hardness assumptions to the overall system.

We however want to note, that with the introduction of a signature-based authentication mechanism, the question arises on how to authenticate the other peer's public key. Note though, that even if all public keys are shared a priori, the complexity is reduced to n keys instead of n^2 pre-shared keys. With the introduction of a Public Key Infrastructure (PKI) such as PKIX [CSF+08], the amount of pre-installed public keys that then serve as certificate authority (CA) can be drastically reduced. In a setting with only one provider, this can be a single CA. With more providers, various different scenarios can be considered with one external CA or multiple CAs where, for example, each provider handles the certification of the public keys used by their network components.

For QKD networks with trusted networks, we note that all trusted nodes have access to the QKD key. In the HAKE security model, we thus need to assume that CompromiseSK has been queried and therefore the security of Muckle+ solely relies on the security of the KEM and signature scheme. To achieve fault tolerance in such a setting, we can consider multipath QKD systems that apply a typical secret-sharing-based approaches, e.g., [KGSR02,FFGV07]. Thereby, the QKD key is shared on the initiator side and the shares are transported via mutually disjoint paths in the QKD network to the receiver. Such an approach has been considered to some extent in the literature specifically for QKD networks, e.g., to boost throughput [YLL+21] and with semi-trusted and fully-trusted paths [CCCZ23] to increase the security of the network. The latter focuses specifically on the routing algorithms without going into details on the method to share the keys. By applying the techniques, e.g., from [KGSR02] to the QKD keys, and under the assumption that at least one path is non-compromissed or more specifically – similar to the non-collusion in multiparty computation systems – that none of the nodes on disjunct paths collude, the risk stemming from trusted nodes can be mitigated.

4 Implementation and Evaluation

In order to evaluate the performance of Muckle+ in practical application, we implemented a prototype of the protocol. This prototype was implemented in Python using bindings[7] of `liboqs` [SM16] for the support of post-quantum prim-

[7] `https://github.com/open-quantum-safe/liboqs-python`

itives and the `cryptography`[8] module for all classically-secure schemes. As displayed Figure 3, the Muckle+ protocol operates on the application layer. The quantum key material is fetched by all endpoints by their respective key managements services (KMS) that provide key material to applications via the interfaces from ETSI GS QKD 014 [ETS19].

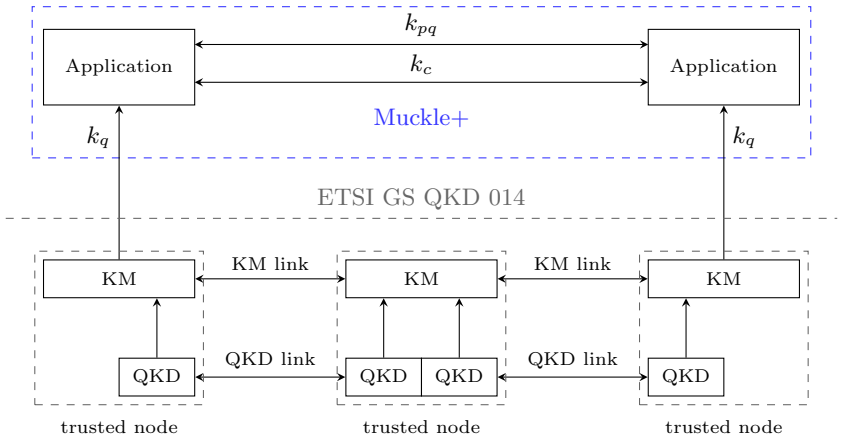

Fig. 3. Architecture of a Muckle+ Implementation with a single intermediate node.

While the Muckle+ protocol allows for server-only, as well as mutual authentication, we benchmarked the implementation with mutual authentication. Authentication of both parties is achieved through the use of hybrid certificates containing both post-quantum and classical long term public keys. Certificates were signed with classical (EdDSA [BDL+11]) and post-quantum signatures. We assumed a 2-tier certificate hierarchy to simulate a PKI hierarchy for Muckle+ closer to the current practice on the web, e.g., similar to certificates issued by Let's Encrypt [ABC+19].

Our Muckle+ implementation was set up using a small network with three QKD links offering two mutually disjoint paths between endpoints. Initiator and responder of the protocol were executed on a notebook running Windows 10 with an Intel i5 2.60GHz CPU and 8 GB of RAM. Several instantiations of the protocol using different post-quantum KEMs and signature schemes were tested, resulting in the execution times displayed in Figure 4 for directly linked nodes. For all executions of the protocol, the remaining primitives have been instantiated with X25519 [Ber06] as KEM_c, HKDF-SHA2 as PRF and HMAC-SHA2 as MAC.

As there was only marginal difference in execution time between the initiator and responder, time values in Figure 4 reflect the runtime for the initiator. Our experiments showed, that for the majority of tested schemes, the execution

[8] `https://pypi.org/project/cryptography/`

time for a single Muckle+ stage ranged from 0.4 to 1.6 seconds. An average of ≈ 0.3 seconds of this runtime can be attributed to the retrieval of the QKD key. Hence, we could demonstrate, that the determining factor in the performance of the Muckle+ protocol is the key rate of the QKD link. An exception to this observation are the s-robust SHA256 and SHAKE variants of SPHINCS+ where the slower runtime is attributed to the lack of implementations making use of the AVX2 instruction set in `liboqs` on Windows. Hence, with an optimized implementation of SHA256 and SHAKE we expect the performance of SPHINCS+ to be closer to those of the other signature schemes.

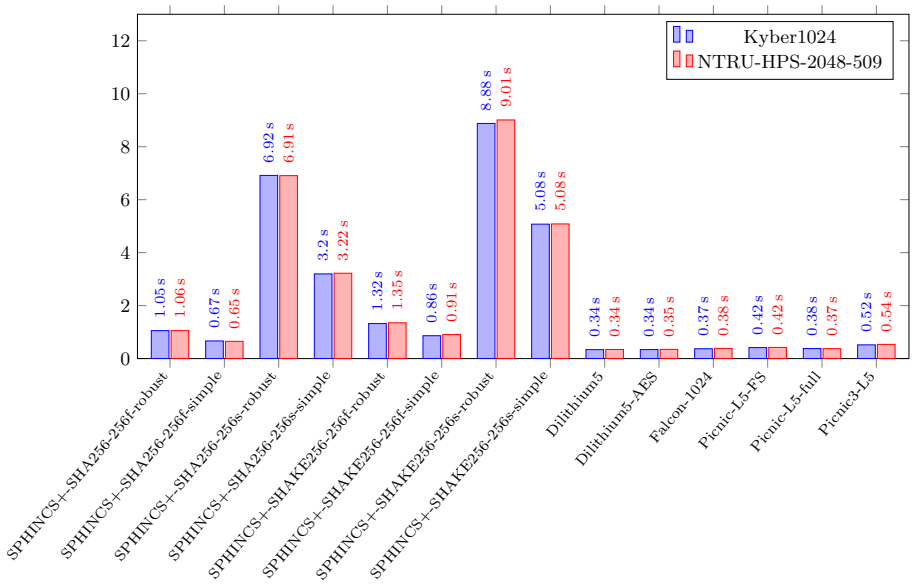

Fig. 4. Execution time of a single Muckle+ stage with mutual authentication. Times are in seconds.

In Figure 5, the results of the same experiment with a multi-path setup are depicted. The additional delay is caused by the intermediate nodes fetching additional key material in a serial manner. The overall execution time of the protocol is thus influenced by the slowest path which in our setup corresponds to the longest path. Overall, we can thus conclude that the overhead of our end-to-end secure protocol for hybrid networks is mainly influenced by the performance of the key rate provided by the QKD network.

5 Conclusion and Outlook

With Muckle+, we extend the hybrid authenticated key exchange protocol Muckle with signature-based authentication. Thereby, we are able to provide both certificate-

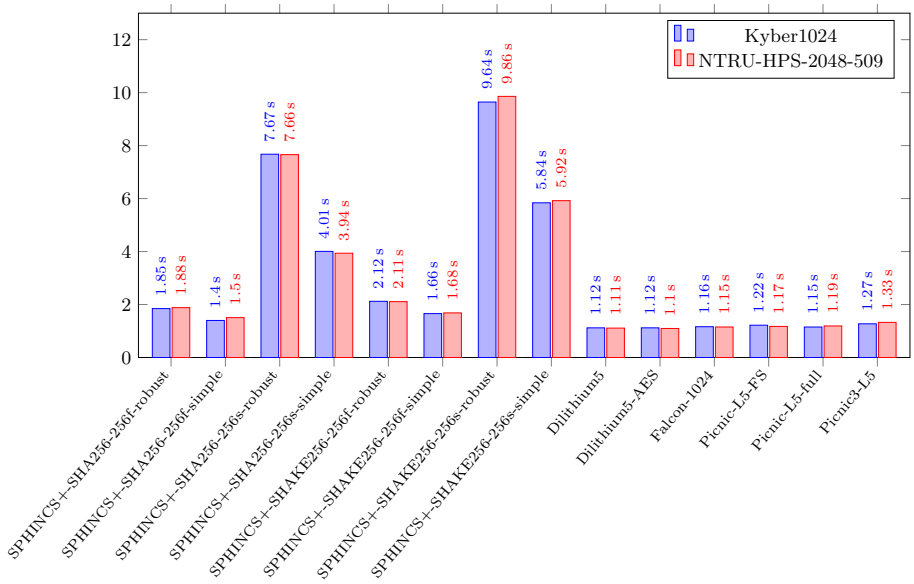

Fig. 5. Execution time of a single Muckle+ stage with mutual authentication. Times are in seconds.

based mutual or unilateral authentication depending on the intended use-case. Our implementation and evaluation of the protocol within a small QKD network demonstrates its practical feasibility. With our intended message flow of the protocol, Muckle+ may be integrated in typical scenarios where especially the authenticity of the responder is essential. We however note that there might be other trade-offs in the order and structure the messages if different privacy properties are required for an application, e.g., SIGMA with responder privacy or a protocol with forward privacy [SSL20,RSW21].

Note also that the Muckle+ protocol offers features that are interesting for a wide range of applications in a similar setting as found in many of today's applications. Indeed, if one considers classical client-server uses on the web, it is expected that one can connect to almost any server on the network without additional configuration. Hence, handling shared state such as the pre-shared keys is not desired to scalability issues as well as the out-of-band communication. Considering more high-level use-cases that are envisioned in EuroQCI where network-wide key management systems will provide QKD keys to security applications, ensuring authenticity with certificate-based mechanisms will provide better scalability especially considering that nowadays process for certificate management can be fully automated [ABC+19].

Acknowledgements. The authors want to thank Christian Rechberger and Felix Wissel for insightful discussions, and Florian Kutschera for helping with the setup of the QKD devices. This work received funding from the Austrian Re-

search Promotion Agency (FFG) under grant agreement number FO999886370 ("QKD4GOV"), from the European Defence Industrial Development Programme (EDIDP) under grant agreement number SI2858093 ("DISCRETION"), and from Digital Europe Program under grant agreement number 101091642 ("QCI-CAT").

References

ABB+14. Romain Alléaume, Cyril Branciard, Jan Bouda, Thierry Debuisschert, Mehrdad Dianati, Nicolas Gisin, Mark Godfrey, Philippe Grangier, Thomas Länger, Norbert Lütkenhaus, Christian Monyk, Philippe Painchault, Momtchil Peev, Andreas Poppe, Thomas Pornin, John G. Rarity, Renato Renner, Gregoire Ribordy, Michel Riguidel, Louis Salvail, Andrew Shields, Harald Weinfurter, and Anton Zeilinger. Using quantum key distribution for cryptographic purposes: A survey. *Theor. Comput. Sci.*, 560:62–81, 2014.

ABC+19. Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth D. Schoen, and Brad Warren. Let's encrypt: An automated certificate authority to encrypt the entire web. pages 2473–2487, 2019.

BBC+21. Ayan Biswas, Anindya Banerji, Pooja Chandravanshi, Rupesh Kumar, and Ravindra P. Singh. Experimental side channel analysis of bb84 qkd source. *IEEE Journal of Quantum Electronics*, 57(6):1–7, 2021.

BBF+19. Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. pages 206–226, 2019.

BCNS15. Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. pages 553–570, 2015.

BDH11. Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. pages 117–129, 2011.

BdK+21. Carsten Baum, Cyprien de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. pages 266–297, 2021.

BDL+11. Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. pages 124–142, 2011.

Ber06. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. pages 207–228, 2006.

Beu22. Ward Beullens. Breaking Rainbow takes a weekend on a laptop. Cryptology ePrint Archive, Report 2022/214, 2022. `https://eprint.iacr.org/2022/214`.

BFG19. Jacqueline Brendel, Marc Fischlin, and Felix Günther. Breakdown resilience of key exchange protocols: NewHope, TLS 1.3, and hybrids. pages 521–541, 2019.

BHK+19. Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. pages 2129–2146, 2019.

BL15. Mihir Bellare and Anna Lysyanskaya. Symmetric and dual PRFs from standard assumptions: A generic validation of an HMAC assumption. Cryptology ePrint Archive, Report 2015/1198, 2015. `https://eprint.iacr.org/2015/1198`.

BMP00. Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. pages 156–171, 2000.

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. pages 139–155, 2000.

BR95. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. pages 57–66, 1995.

CCCZ23. Liquan Chen, Jing-Qi Chen, Qian-Ye Chen, and Yong-Li Zhao. A quantum key distribution routing scheme for hybrid-trusted QKD network system. *Quantum Inf. Process.*, 22(1):75, 2023.

CCG+18. Sergiu Costea, Marios O. Choudary, Doru Gucea, Björn Tackmann, and Costin Raiciu. Secure opportunistic multipath key exchange. pages 2077–2094, 2018.

CDG+17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. pages 1825–1842, 2017.

CF12. Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. pages 734–751, 2012.

CSF+08. David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *RFC*, 5280:1–151, 2008.

DFGS21. Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol. 34(4):37, October 2021.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. 22(6):644–654, 1976.

DHP20. Benjamin Dowling, Torben Brandt Hansen, and Kenneth G. Paterson. Many a mickle makes a muckle: A framework for provably quantum-secure hybrid key exchange. pages 483–502, 2020.

Don17. Jason A. Donenfeld. WireGuard: Next generation kernel network tunnel. 2017.

DvOW92. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.*, 2(2):107–125, 1992.

ETS19. ETSI. Quantum key distribution (qkd): Protocol and data format of rest-based key delivery api, 2019.

FFGV07. Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, and S. Harsha Vardhan. Towards optimal and efficient perfectly secure message transmission. pages 311–322, 2007.

HAD+22. Bruno Huttner, Romain Alléaume, Eleni Diamanti, Florian Fröwis, Philippe Grangierand Hannes Hübel, Vicente Martin, Andreas Poppe, Joshua A. Slater, Tim Spiller, Wolfgang Tittel, Benoit Tranier, Adrian Wonfor, and Hugo Zbinden. Long-range qkd without trusted nodes is not possible with current technology. *npj Quantum Information*, 8(1):1–5, 2022.

HBG+18. Andreas Hülsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: extended merkle signature scheme. *RFC*, 8391:1–74, 2018.

HKSU20. Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. pages 389–422, 2020.

HNS+21. Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. Post-quantum WireGuard. pages 304–321, 2021.

IT21. Jana Iyengar and Martin Thomson. QUIC: A udp-based multiplexed and secure transport. *RFC*, 9000:1–151, 2021.

Kau05. Charlie Kaufman. Internet key exchange (ikev2) protocol. *RFC*, 4306:1–99, 2005.

KGSR02. M. V. N. Ashwin Kumar, Pranava R. Goundan, K. Srinathan, and C. Pandu Rangan. On perfectly secure cmmunication over arbitrary networks. pages 193–202, 2002.

KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. pages 525–537, 2018.

KL14. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

Kra03. Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. pages 400–425, 2003.

KSL+19. Krzysztof Kwiatkowski, Nick Sullivan, Adam Langley, Dave Levin, and Alan Mislove. Measuring tls key exchange with post-quantum kem. Workshop Record of the Second PQC Standardization Conference, 2019. `https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/kwiatkowski-measuring-tls.pdf`.

Lan16. Adam Langley. Cecpq1 results. Blog post, 2016. `https://www.imperialviolet.org/2016/11/28/cecpq1.html`.

LKZC07. Jin Li, Kwangjo Kim, Fangguo Zhang, and Xiaofeng Chen. Aggregate proxy signature and verifiably encrypted proxy signature. pages 208–217, 2007.

LRW+17. Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan R. Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. The QUIC transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, pages 183–196. ACM, 2017.

Mau93. Ueli M. Maurer. Protocols for secret key agreement by public discussion based on common information. pages 461–470, 1993.

MNR+20. Miralem Mehic, Marcin Niemiec, Stefan Rass, Jiajun Ma, Momtchil Peev, Alejandro Aguado, Vicente Martín, Stefan Schauer, Andreas Poppe, Christoph Pacher, and Miroslav Voznák. Quantum key distribution: A networking perspective. *ACM Comput. Surv.*, 53(5):96:1–96:41, 2020.

MSU13. Michele Mosca, Douglas Stebila, and Berkant Ustaoglu. Quantum key distribution in the classical authenticated key exchange framework. pages 136–154, 2013.

PST20. Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in TLS. pages 72–91, 2020.

Res18. Eric Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018.

RK11. Stefan Rass and Sandra König. Indirect eavesdropping in quantum networks. In *ICQNM 2011: The Fifth International Conference on Quantum, Nano and Micro Technologies*, 10 2011.

RKJ+21. Leila Rashidi, Daniel Kostecki, Alexander James, Anthony Peterson, Majid Ghaderi, Samuel Jero, Cristina Nita-Rotaru, Hamed Okhravi, and Reihaneh Safavi-Naini. More than a fair share: Network data remanence attacks against secret sharing-based schemes. 2021.

RS10. Stefan Rass and Peter Schartner. Multipath authentication without shared secrets and with applications in quantum networks. In Hamid R. Arabnia, Kevin Daimi, Michael R. Grimaila, George Markowsky, Selim Aissi, Victor A. Clincy, Leonidas Deligiannidis, Donara Gabrielyan, Gevorg Margarov, Ashu M. G. Solo, Craig Valli, and Patricia A. H. Williams, editors, *Proceedings of the 2010 International Conference on Security & Management, SAM 2010, July 12-15, 2010, Las Vegas Nevada, USA, 2 Volumes*, pages 111–115. CSREA Press, 2010.

RSW21. Sebastian Ramacher, Daniel Slamanig, and Andreas Weninger. Privacy-preserving authenticated key exchange: Stronger privacy and generic constructions. pages 676–696, 2021.

SM16. Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. pages 14–37, 2016.

SSL20. Sven Schäge, Jörg Schwenk, and Sebastian Lauer. Privacy-preserving authenticated key exchange and the case of IKEv2. pages 567–596, 2020.

SSW20. Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. pages 1461–1480, 2020.

YLL+21. Xiaosong Yu, Xiang Liu, Yuhang Liu, Avishek Nag, Xingyu Zou, Yongli Zhao, and Jie Zhang. Multi-path-based quasi-real-time key provisioning in quantum-key-distribution enabled optical networks (qkd-on). *Opt. Express*, 29(14):21225–21239, Jul 2021.

Zha16. Yunlei Zhao. Identity-concealed authenticated encryption and key exchange. pages 1464–1479, 2016.